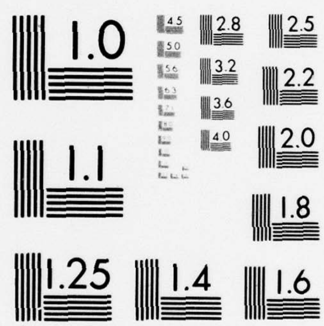
AD-A062 791 TEXAS UNIV AT AUSTIN COLL OF ENGINEERING F/G 9/2
PROGRESS REPORT OF THE WORKING GROUP OF THE CONFERENCE ON COMPU--ETC(U)
OCT 77 N00014-78-M-0033

UNCLASSIFIED NL

1 OF 1

AD
A062791

END
DATE
FILMED
3-79
DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(12)

THE UNIVERSITY OF TEXAS AT AUSTIN
COLLEGE OF ENGINEERING
AUSTIN, TEXAS 78712

Department of Electrical Engineering
Dept. Ph. 512—471-1851, Chairman 471-4262

June 18, 1978

My Dear Colleagues

As you can see from the enclosed memo, we are coming out of hiding. Actually, the information was available last February, but the money to have it printed was not processed, and it took until now to clear up the red tape.

Since the memo was written, we had another short meeting in May, and we expect to have another in August. So if you have any comments or suggestions on the enclosed material, please return it to me by July 15.

I would like to apologize for my tardiness in getting this memo to you, but I would like to express my appreciation for the effort the working group has put out in getting this memo together. I think it is a big step towards our preliminary consensus language.

Sincerely

G.J. Lipovski
Chairman, Executive Committee
Conference on Digital Hardware Languages

BALLOT 31

To be returned to: G. Jack Lipovski
Department of Electrical Engineering
The University of Texas at Austin
Austin, Texas 78712

To be returned by July15, 1978

Please sign your name: ______________________________

	YES	NO	ABS	YIELD
1. Mario Barbacci will replace Yaohan Chu on the Working Group that is preparing a preliminary CONLAN report.	___	___	___	___

Please get me in touch with Yaohan to get more information on the IFIP technical committee on Digital Systems Design /__/

Please use the space below to comment on the enclosed memos, etc.

Memo 3.21

With great pleasure, I offer you a report on the progress of the Conference on Digital Hardware Languages. Firstly, I want to acknowledge support from Dr. Lennie Haynes of the Office of Naval Research, who provided funds for a meeting of the working group, January 12-15 at Fred Hill's site, the University of Arizona. This grant for the meeting also contained publication money, which was used to pay for the reproduction and mailing of the enclosed report. Secondly, I can report that I have committments by phone for the sponsorship of the next two meetings of the working group. So it looks like our financial problems for supporting the working group are under control. Thirdly, Fred Hill has prepared a report, which has been approved by the working group for circulation to the members of the conference and to the general public. This report is enclosed as memo 3.20. While it is premature to publish this report in trade journals or proceedings, we think it is appropriate to make it available to anyone who is interested. Importantly, the working group asks you to inspect it and offer you evaluations and suggestions of it to them, so they can tune up their proposal. When their work is done, they will send a report to the Executive Committee and the conference as a whole including you, for detailed analysis and revision. This should happen in about a year. Meanwhile, regard this report as a preliminary indication of where the working group is headed, and offer your recommendations to them in this light. If you have suggestions or criticisms, send them to me and I will forward them to the group.

As I scan the report, I am impressed with the thoroughness and soundness of the approach that the working group is taking. I am also aware that other members of the conference would prefer other approaches. Indeed, I remind you that I am not a member of the working group and have not attended any of their meetings. The language they are coming out with is quite different than the one I had in mind. I think it is better. You may find yourself in the same situation after reading the report. Moreover, whereas, in some countries, you drive on the left side of the road and in others you drive on the right side, the two rules are equally correct provided you don't mix them. The working group's language is beautifully consistent. Currently, with our myriad of hardware description languages, we are driving on every side of the road. I strongly believe that, whatever consistent rules we adopt, even if I have to change my habits and preferences, we have to aim at consistency. Therefore I am committed to support the language that the conference finally adopts. I hope, furthermore, that the language the working group comes out with has a good chance of becoming the language that we finally adopt, after improvements are made that the members of the conference suggest. On this basis, I encourage you to thoroughly study the report and give your reaction to it to the working group, by sending it to me.

The working group, which was created by the Executive Committee, requests a change which requires a vote by the Executive Committee, that by our procedures is voted on by the entire conference. Firstly, Yaohan has become extremely busy with other activities, one of which I will discuss shortly. He has asked to be relieved of his responsibilities in the working group. Secondly, Yaohan would then discontinue his participation in the preliminary CONLAN working group to establish this new group, and simultaneously Mario Barbacci would like to join the working group, I propose the second question on the ballot. Mario brings a lot of expertise on hardware description languages, and especially on ISP, so that his

background is exceptional. Mario attended the meeting of the working group at the University of Arizona as an observer. The working group has sent me a strong recommendation that Mario be officially appointed to it. I think the working group is fortunate that, while it is losing Yaohan's capable participation, it can gain Mario's.

You may recall that Robert Piloty has been charged by the Conference to seek some form of connection between IFIP (The International Federation for Information Processing) and the Conference. He has initiated a technical committee (TC10) in IFIP that will study Digital Systems Design. One of the three subgroups of TCID will study digital hardware languages. This subgroup may indeed overlap our conference, but is administratively separatè from it. We feel that this subgroup and the technical committee may offer a vehicle to promote CONLAN in the international community at the appropriate time. Yaohan Chu is the U.S. representative on TC10, and is agressively pursuing it. This is one of the main reasons why Yaohan would like to end his participation in the working group that is developing preliminary CONLAN. Yaohan has asked me to help seek out U.S. members for this technical committee in general, and for the subgroup on hardware description languages in particular. I think that our Conference should be made aware of this activity and should be enco raged to participate. So if you would like to find out more about it, either contact Yaohan directly or check the appropriate place on the ballot and I will get your name to Yaohan so he can contact you.

I encourage you to send me your answers to the questions on ballot 31. While the binding decisions of the conference depend on the vote of the Executive Committee, the votes of all the members help check the Executive Committee and have prompted a requiest for a second vote wherever the conference vote differed from the Executive Committee vote. Moreover, all the intense activity of the working group has been done behind closed doors. Your participation in responding to a ballot like this is needed to reassure the working group that you are still there - that its results will be eagerly sought by the conference as a whole, and should prompt them to hurry up their work. Space is provided so that if you wish to find out more about the IFIP technical committee TC10, you can mark the box. We will try to get more information to you. Finally, the remaining space on the ballot should invite you to send back comments on memo 3.20. Be assured that, even if the comments are just handwritten, I will try to decipher then to get them typed up for the working group. But extensive comments on additional sheets will also be welcome, of course.

~~Memo 3.20~~

6

Progress Report of the Working Group of the Conference on Computer Hardware Description Languages

11 20 Oct [redacted]77

12 27p.

15 [redacted]t N00014-78-M-0033

Chairman: Robert Piloty

Members: Dominique Borrione
Yaohan Chu
Don Dietmeyer
Fred Hill
Pat Skelly

406 085

GVL

I Overview

The first workshop on hardware description languages was organized by Professor Jack Lipovski and held at Rutgers University in September 1973. The forty attendees present established the "Conference on CHDL" and approved an executive committee with the goal of preparing a consensus language referred to herein as CONLAN.

Phase 1 of the activities consisted of the distribution of 14 memos and 8 ballots where members voted on establishing committees and general language concepts. A second work shop was held in Darmstadt, Germany in August 1974. The special December 1974 issue of COMPUTER was derived largely from papers presented at this meeting. Phase 2 continued the activities with 37 memos and 25 ballots as financial support had not yet been located.

The third meeting of the conference was held in September 1975 in New York. Formal proceedings were published with IEEE and ACM support. The feeling expressed was that development by the entire conference by mail was far too slow. A working group with Robert Piloty chairman, Dominique Borrione, Pat Skelly, Yaohan Chu, Fred Hill and Don Dietmeyer was established and charged with preparing a preliminary language.

The first set of meetings of the working group was held under the sponsorship of Bell Northern Research in Ottawa, Canada from June 11 to June 14, 1976. At this meeting language guidelines and a tentative syntax for individual

statements were accepted, and the concept of a base language from which higher level languages could be derived was first introduced. This concept was tentatively accepted by the group and individual group members between meetings to expand this notion.

The second set of meetings of the working group was held under the sponsorship of Sperry Univac at Valley Forge Pennsylvania from Jan. 6 through Jan. 9, 1977. At these meetings Robert Piloty presented a series of six memo detailing a language module structure for extending base CONLAN. These proposals were based in part on the work of B. Liskov and S. Zilles of MIT [1,2,3,4]. Much of the time at the meetings was devoted to examination and refinement of these proposals. Following these meetings individual group members considered problem areas suggested by the Piloty memos, with particular attention to an interpretation of time applicable at all language levels.

The third set of working group meetings were held in Toronto, Canada on July 31, August 1, and August 2, 1977. At this meeting the definition of base CONLAN and mechanisms for its extension to higher levels were further formalized. A general approach to the treatment of time was presented. In addition a partial top down syntax and the notion that such a syntax would serve as a guide for and perhaps bounds on the evolution of new semantic constructs from base CONLAN was accepted. At this meeting, an alternative approach was proposed by Yaohan Chu. Instead of a mathematical approach, he

proposed an engineering and experimental approach. This approach which is top-down, was used to describe two simple microcomputers. Prof. Chu's proposed language constructs also permits a family of languages for use at various levels of detail.

The working group is committed to meeting at least twice each year until releasable versions of base CONLAN, the language extension mechanisms, and sample languages at various user levels have been approved. Between meetings the group will carry on its work by circulating memos through the mail.

Due to space limitations and the inevitability of change within the tentatively accepted portions of CONLAN we shall make no attempt to detail a complete langauge in this document. Rather a sampling of constructions at various langauge levels will be presented with the goal of introducing the reader to the working group's approach.

II Guidelines

The following have been accepted as guidelines for the consensus language.

A. CONLAN must support design, description, and simulation of at least the following classes of systems: gate network, register networks, processors, memories, processor systems. Each class has been fully defined.

B. Any system may be displayed via either (a) a network structure description or (b) a behavior description.

C. CONLAN is to service:

~~1~~. Computer architects and logic designers for purposes of trade-off exploration and optimization, design verification, and design documentation.

next page

~~2~~. Systems, micro, and applications programmers,

~~3~~. Electronics production engineers, and

~~4~~. Maintenance engineers.

D. CONLAN syntax and semantics must support:

1. Well-defined descriptions
2. Machine parsing, interpretation and simulation with error detection (strong typing has been adopted)
3. Comprehension of complex system structure and function
4. Division of design efforts
5. Control over the level of abstraction at which subsystems are described.
6. Simulation control

E. CONLAN will be evaluated in terms of benchmarks such as: standard function declarations, time operator declarations, integrated circuit descriptions (long list, including microprocessors), design descriptions (another long list including a multiprocessor system).

III The Base CONLAN Concept

The above guidelines for a consensus hardware description language calls for a language capable of representing hardware at several distinct levels of detail. At the lowest level gate networks in which even flip-flops are non-primitive elements must be described in CONLAN. In contrast algorithms expressed in assembly language or by an abstract real time independent

language level can represent hardware and must interact with lower level hardware descriptions. Thus, these higher level representations must also be expressible in CONLAN. Some potential CONLAN levels might not be envisioned at present.

The requirements of this range of language levels would seem to suggest not a single consensus language but a family of languages. If the members of this language family were to be independent, we would have fallen seriously short of the goal of a universal communications medium understandable by all practitioners in digital hardware. Thus the various language levels must <u>share a common basic syntax</u>. The basic syntax will be augmentable by specific constructs with their own semantic interpretation, as required by higher language levels. Representating a digital system during the design process at different levels of abstraction requires that semantics be related from level to level. Now consider language implementation or system simulation. This implies the interaction of hardware segments described by more than one language level (e.g. some as actual hardware, others as the interface behavior of grey boxes). This interaction is possible only if the two language levels are based on a <u>common interpretation of time.</u> Similarly a mechanism for translation between data types must be defined in terms of a common semantic base for the two languages.

The foregoing are samples of the considerations which led to the proposal of the <u>base CONLAN</u> concept. The heart of

base CONLAN is a definition mechanism which allows the repertoire of available data types and associated operations to change in moving up from level to level. Data types useful at one language level would be defined in terms of those of lower language levels. The data types appearing in any subset of the CONLAN family would thus be rooted in a common semantic base which could be implemented in a compiler/simulator designed to process this set of language levels.

The vehicle by which new types are defined in terms of existing types is the <u>type module</u>. Several type modules may be tied together in a <u>language module</u>, describing all data types and operations available in a CONLAN sublanguage. Informally we present the following as a simple example of a type module which might appear at some point in the development of the language. This module defines the data type <u>boolean</u> and the operations OR, and NOT in terms of the data type <u>ternary</u> which has previously been defined. As might be expected, the prior definition of the ternary OR operation, <u>tor</u>, and ternary NOT, <u>tnot</u>, is more complex relying on the set constructors of primitive set CONLAN to be presented in the next section.

```
type boolean
    reflan ternary, primitive set conlan endreflan
    rep {0:1}
    function  OR(x,y:boolean):boolean
              = Tor(x,y) end function
```

```
    function not(x,y:boolean):boolean
             = Tnot(x,y) end function
end type
```

Underlined words are reserved words whose definitions are part of primitive set CONLAN. Following the reserved word reflan is a list of all type and language modules whose defined data types, functions, etc. may be used in definitions in the language module "boolean." In this case the language module "primitive set conlan" and the type module "ternary" are assumed to have been defined previously. All definitions pertaining to a single data type are bracketed by the reserved words type/endtype. The data type "boolean" is represented by the set of objects {0,1} available in primitive set conlan. The definition of each function applicable to "boolean" is bracketed by function/endfunction. The format "(x,y:boolean): boolean" indicates that the arguments x,y are drawn from the set "boolean" as defined by "rep {0,1}" and that the range of functional values is also the set boolean.

The overall structure of CONLAN is illustrated in Fig. 1. From the baselanguage, L_0, a family of application languages L_1, L_2, L_3, L_4, L_5, and L_6 is derived. A description may then be written in terms of any one of the application languages.

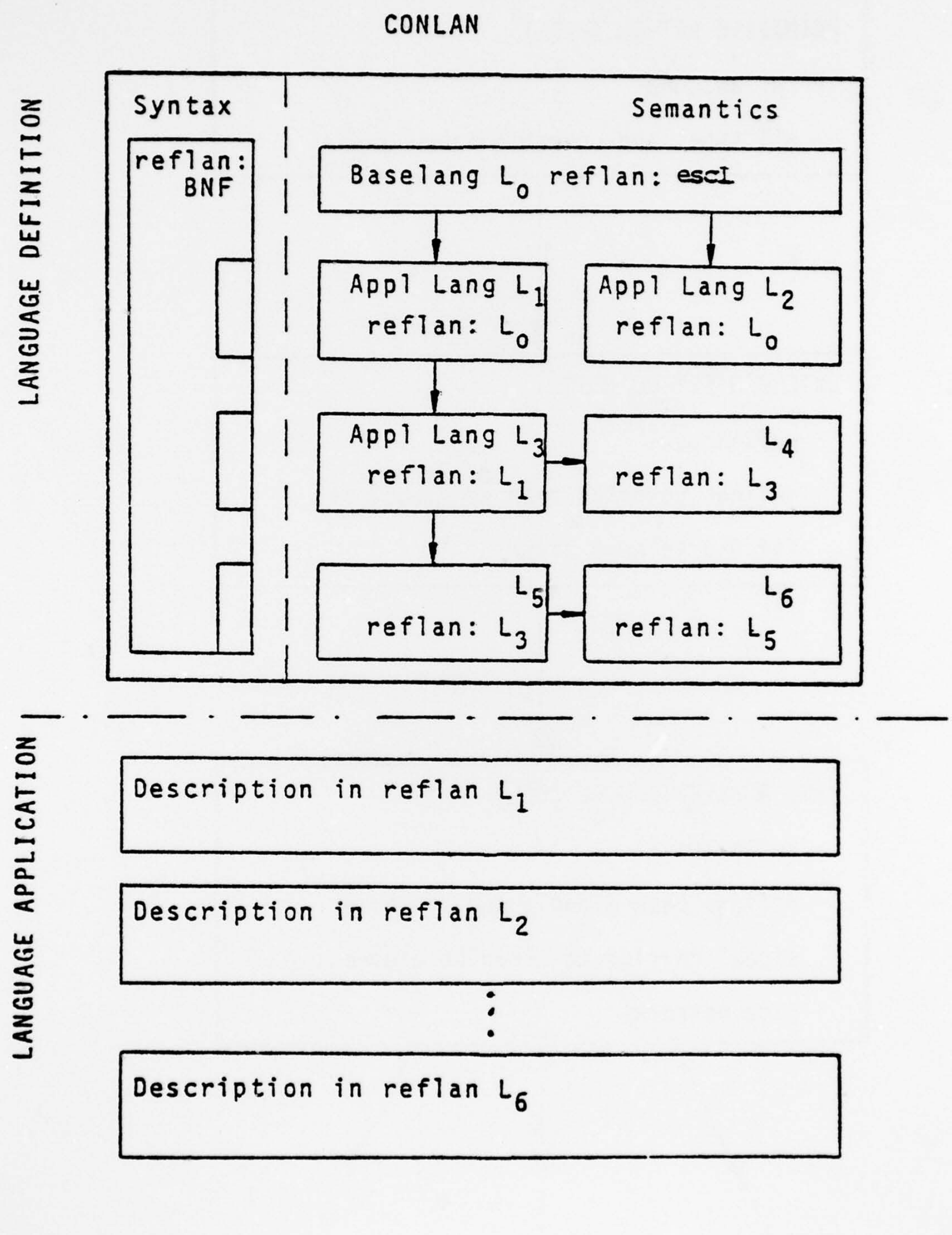


Figure 1 Textstructure of Selfdefining Language Family

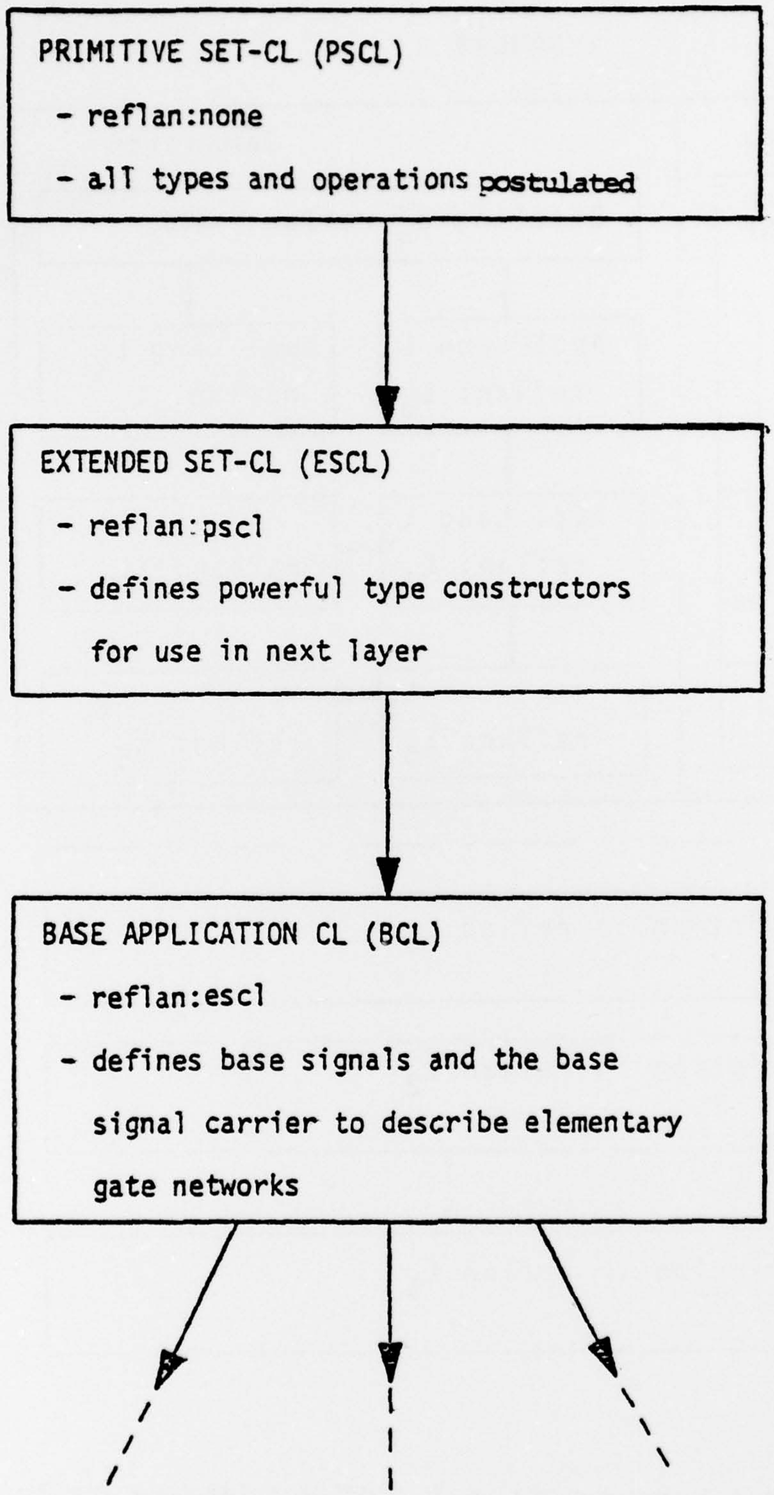


Figure 2 Groundlayers of CONLAN

The base language of Fig. 1 is not the most primitive level of CONLAN. Figure 2 depicts two lower levels. The most basic level, primitive set CONLAN (PSCL), postulates only set constructors and a predicate format. The notion of a carrier together with a representation of time are postulated at the second language level, ESCL. Everything in the remaining levels of CONLAN can be defined in terms of these basic constructs.

The reader may view the structure of figures 1 and 2 as indicating a complexity which would seem forbiding to a typical user of a hardware description languages. Most language users will, however, be concerned with only one or two application level languages and associated software. Those who write software for the processing of application level languages will have occasion to examine referenced language modules. It is likely that only researchers in the field will have occasion to write language modules.

IV Syntax of CONLAN

The definition process described in the previous section serves only to insure a logical relationship between data types and operations defined at the various language levels. It places no constraints on the format of expressions at any level, nor does it establish any bounds on the expansion of the language. This is the role of the standard CONLAN syntax, parts of which have been tentatively adopted. The outer most units in CONLAN are either descriptions of

hardware or language modules for expanding CONLAN semantics. This is the startingpoint for the partial syntax given below.

It has been agreed that the BNF productions would be shortened by using ⌈<x>⌋ to indicate the optional inclusion of <x>. Similarly ≺<y>≻ will represent the inclusion of <y> zero or more times.

```
<whole>::= <description> | <language module group>
<lang_module_group>::= <lang_mod> | <lang_mod> <lang_
   module_group>
<lang_mod>::= langmod <mod_id>⌈(<mod par>)⌋ <lang_
   mod_body> endlangmod
<mod_id>::= ¢ typical_id¢
<mod_par>::= ¢ to be defined ¢
<lang_mod_body>::= reflan <lang_id_list>endreflan
  <type_group> ≺<op_group>≻
<lang_id_list>::= <mod_id> ≺<lang_id_list>≻ |
     <type_id> ≺ <lang id list>≻
    ¢ an unordered list of mod_id and lang_id ¢
<type_group>::= <carried_over_type> ≺ <type_group>≻ |
   <new type> ≺ <type group>≻
<carried_over_type>::= carry <list_of_type_id> |
   carryall
<list_of_type_id>::= ¢ to be defined ¢
<new type>::= type <type_id> ⌈ <type param list>⌋
   <type body>endtype
<type_id>::= ¢ to be defined ¢
<type_param_list>::= ¢ to be defined ¢
<type_body>::= rep <set_expression> endrep ≺<list-
   of-operations>≻
<set expression>::= ¢ to be defined ¢
<list_ of_operations>::= <function_def> |
                         <procedure_def> |
                         <action_def>|
                         <activity_def>
```

The above BNF list expands the language module consistant with the illustration of the module, "boolean" in the previous section. To avoid unduly constraining the functioning of the working group the syntax has not yet been completely formalized at the intermediate levels. This is evidenced by the appearance of the comment ¢ to be defined ¢. A partial continuation of the syntax for a <description> is given below.

<description>::= description [<descr_id>] <decl_part> <behavior_part> end description

<decl_part>::= <reflan_decl> <declaration> ≺<declaration>≻

<behavior_part>::= <segment> ≺<segment>≻

<segment>::= segment [<segment_id>] <object_declarations> <program>

<declaration>::= <template_decl> | <operation_decl>| <object_decl>

<operation_decl>::= <procedure_decl> | <function_decl> | <action_decl>

<object_decl>::= <carrier_decl> | <activity_instance_declaration>

It can be seen that the term <declaration> should be interpreted as not only stating the existence of items, but as providing templates of items as well. Thus an <operation-decl> might define a frequently used procedure or subroutine. The simplest declaration would be a carrier declaration which could for example, be a memory element or flip flop at some language levels.

At the other end of the syntax spectrum, the use of

<table>
<tr><th></th><th>OPERATOR</th><th>NOTATION</th><th>DOMAIN</th></tr>
<tr><td>1</td><td>AND</td><td>∧ and</td><td rowspan="2">binary
bit × bit → bit
bool × bool → bool</td></tr>
<tr><td>2</td><td>OR</td><td>∨ or</td></tr>
<tr><td>3</td><td>NOT</td><td>¬ not</td><td>unary bit → bit
bool → bool</td></tr>
<tr><td>4</td><td>is less than</td><td>< lss</td><td rowspan="4">binary
int × int → bool</td></tr>
<tr><td>5</td><td>is greater than</td><td>> gtr</td></tr>
<tr><td>6</td><td>is not less than</td><td>≥ geq</td></tr>
<tr><td>7</td><td>is not greater than</td><td>≤ leq</td></tr>
<tr><td>8</td><td>is equal to</td><td>= eq</td><td rowspan="2">binary
int × int → bool
char × char → bool
bool × bool → bool
bit × bit → bit</td></tr>
<tr><td>9</td><td>is not equal to</td><td>≠ neq</td></tr>
<tr><td>10</td><td>add</td><td>+ plus</td><td rowspan="6">binary int × int → int</td></tr>
<tr><td>11</td><td>substract</td><td>− minus</td></tr>
<tr><td>12</td><td>multiply</td><td>× mult</td></tr>
<tr><td>13</td><td>divide</td><td>÷ div</td></tr>
<tr><td>14</td><td>residue</td><td>| mod</td></tr>
<tr><td>15</td><td>power</td><td>↑ pow</td></tr>
<tr><td>16</td><td>minus</td><td>− minus</td><td rowspan="2">unary int → int</td></tr>
<tr><td>17</td><td>absolute value</td><td>abs</td></tr>
<tr><td>18</td><td>reduction</td><td><oper> / red</td><td>unary on int, bool, bit; applies to the first dimension of the operand (the operand may have any number of dimensions)
<oper> is any allowed (depending on the type of the operand) binary operator.</td></tr>
<tr><td>19</td><td>decode</td><td>⊥(X) decode</td><td>bit → posint
high order bits on the left</td></tr>
<tr><td>20</td><td>encode</td><td>⊤(X,N) encode</td><td>posint × posint → bit
the first posint is encoded
the second tells the number of bits of result</td></tr>
<tr><td>21</td><td>delay</td><td>Δ(X,N) delay</td><td>bit × posint → bit
posint tells the number of time units</td></tr>
</table>

Table I Tentative CONLAN Function Symbols

certain basic symbols have been agreed upon for use within the working group. As elements of CONLAN semantics are derived, the associated formats will be made consistant with this standard. Both symbols and reserved words for anticipated CONLAN functions are listed in Table I. The following symbols will be used to specify the updating of carriers.

variable assignment	:=
transfer	←
connection	.=

Dimensions are indicated by square brackets with no restriction on ordering. Indices are unsigned with the following separators.

:ranges
, non contiguous indices
; separation of dimensions

These symbols are similarly used in references to arrays. Concatenation is allowed on any dimension. Periods may be used to indicate catenation. The number of periods indicates the dimension. For example, A..B indicates the catenation of the rows of two arrays.

V Primitive Set CONLAN (PSCL)

The basic sets, set constructors and predicate forms which compose primitive set CONLAN are listed below. The critical notion is that of a data type or type. The only

specific types postulated in PSCL are integers and characters.

1. Types and Classes of Types

   a) primitive elements primel

      elements: signed and unsigned numbers, character set, identifiers (free, reserved, quoted)

      operations: =, ≠

   b) universal type univ

      definition: $x \in$ atype implies $x \in$ utype

      operations: , , =, ≠

   c) universal class of types anytype
      abbreviated any

      definition: x=atype implies $x \in$ any

      operations: $\in$, $\subset$, =, ≠, #(cardinal number)

   d) integers int

      operations: +, −, x, ÷, \| , ↑, <, >, ≤, ≥, =, ≠;

   e) free identifiers ident

      operation: =, ≠

   f) character char

      operations: =, ≠, >, <, ≥, ≤;

2. Basic set constructors

   a) set constructor $\{x_1, x_2, \ldots, x_n$: univ $\}$

   b) tuple constructor $(x_1, x_2, \ldots, x_n$: univ $)$

   c) subset constructor

      all $x \in u$; any with pred (x,a,b,c,) endall

d) element constructor

<u>the</u> x ∈ u: <u>any</u> <u>with</u> pred (x,a,b,c) <u>endthe</u>

e) class constructor

<u>set</u> atype (u,v,w: <u>utype</u>) <u>over</u> u, v <u>with</u>
pred (u,v,a,b) <u>endset</u>

3. Predicate forms

- expressions
- universally quantified predicates:

<u>forall</u> x ∈ u: <u>any</u> <u>is</u> pred (x,a,b,c) <u>si</u>

- existentially quantified predicates

<u>forone</u> x ∈ u: <u>any</u> <u>is</u> pred (x,a,b,c) <u>si</u>

4. Definitions of reserved words

¢ list presently incomplete

See Section II for examples ¢

VI Extensions

We might now envision a single language module representing the development of extended set CONLAN. The definition of the data types positive integers, ternary, and boolean would be part of this module. Also included would be the following data type which greatly increase the facility for expression in CONLAN.

```
type pws (u: any ) rep all x  any with x⊆u endall
                    endtype
type cross (u,v: any) rep set (ℓ,r) with (ℓ∈ u)
                    ∧(r∈ v) end set
    function le (x: cross): u = ℓ endfunction
    function re (x: cross): u = r endfunction
endtype
```

These data types are the usual definitions of powerset and cross product with functions which pick out the left and right member of any ordered pair in a cross product. It is next possible to define the extremely useful data type "map," which is effectively the set of all possible mappings of a domain v into a range u. A variety of functions have been defined on this data type but only one which counts the number of elements in the domain is given here.

```
type map (u,v: any)
     rep all m∈ pws(cross(u,v)) with forall
     x1 and x2∈ m is if le(x1) = le(x2)
     then re(x1) = re(x2) si endall
     function cardom(m: map(u,v.)): posint = #(u) endfunction
               .
               .
               .
end type
```

CONLAN assumes that <u>real</u> <u>time</u> is divided into discrete intervals, the length of which will be specified to the language processing software by the user. The language must allow the possibility of indefinitely long sequences of computations in one real time interval. We say that these computations are accomplished in <u>virtual</u> <u>time</u>. Virtual time will also be divided into discrete intervals. The number of virtual time intervals in each real time interval is indefinite, depending on the processes to be carried out. This leads to the representation of time depicted in Figure 3. Four real

time intervals are shown, the first including 3 virtual time steps, the next three with 2, 6, and 2 steps respectively.

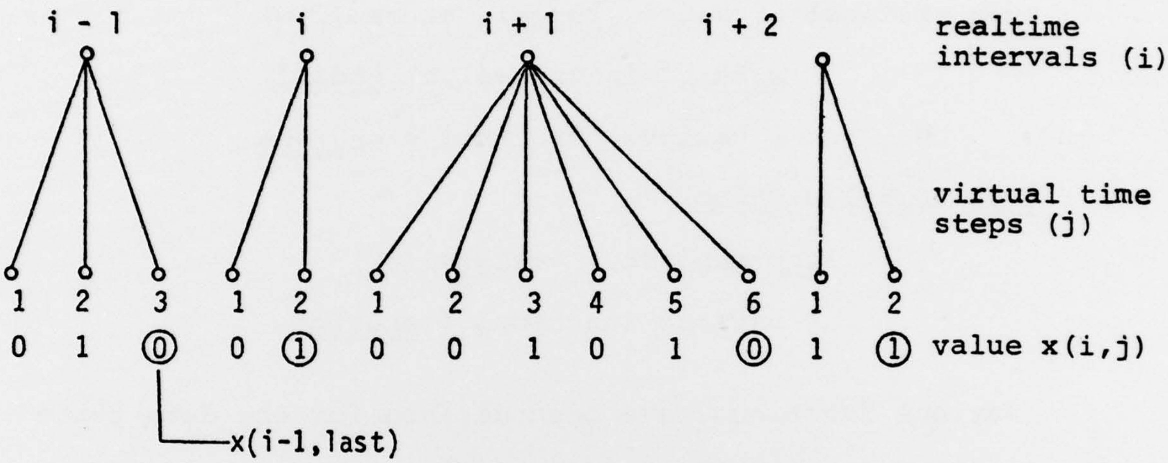


Figure 3 CONLAN Timing

A data type which we shall call "signal" is all possible sequences of values drawn from some data type over real and virtual time. The term "value" represents the signal state at a given point in real and virtual time. A value x at a particular time may be designated as x at i st j with i indicating the real time interval and j the virtual time step.

The following is a formal definition of the data type "vt signal" which contains all sequences of values over a single real time interval. This data type is then used in the formal definition of the data type "signal."

```
type interval (l,n:int) rep all iε int with l≤i≤n
                endall endtype
type intrange rep all interval(l,n) with nε posint
                endall endtype
type vtsignal(u:value)  rep all sε map(r,u)
                with rε intrange(l,n) endall
                ¢ various functions ¢ endtype
type signal(u:value)
              rep map(int,  vtsignal(u))
              ¢ various functions ¢ endtype
```

Various functions have been defined for the data types "vtsignal" and "signal" including delay. These are refered to only by comment in the above abbreviated development. From this point the development continues with the definition of a carrier and leads eventually to the definition of memory elements and a control structure for higher level languages.

A summary of all the data types and the corresponding applicable operations defined in ESCL is given in Table II. The continuation of the definition process through the next level forms base CONLAN. The data types and operations which have thus far been developed for base CONLAN are given in Table III.

types	operations
type	$\in, \subseteq, =, \neq$
primel, utype	{ }, ( )
any	{ }, ( )
pws(u:any)	#(x)
cross(u:any)	ℓe(x), re(x)
map(u,v:any)	x(a), dom(x), rge(x), cardom(x)
list(u:any)	x[i], lgth(x)
vector(u:any;n:posint)	x[i],
value	
bool, tern	$\wedge, \vee, \neg$
int, posint	arithm.op, arithm.rel
char	order relations
ident	$=, \neq$
signal	
signal(u:value)	x at i st j
	delay(x:signal(u);d:posint)
timer	
vtimer	cont(s)
rtimer	cont(t)
carrier	decl(id;ctype)
carrier(u:signal)	npart(x), spart(x),

Table II Survey of ESCL

types	operations
signal	delay(x:(u:signal);d:posint)
sbool,	$\wedge$, $\vee$, $\neg$
stern	$\wedge$, $\vee$, $\neg$
sint	arithm.op., arithm rel.
schar	order relations
timer	
rtimer	cont(t)
carrier	decl(id;ctype)
tml(u:signal)	conn(x;y), outp(x)
type	
vector(u:any;n:posint)	x[i]
ordarray(u:any)	dim(x), bound(x,d)
	x[iℓ1;iℓ2;...], cat(x,y;d)
array(u:any)	dim(x); lb(x,d); rb(x,d)
	x[iℓ1; iℓ2;...], cat(x,y;d)
record(f:fieldlist)	f of x
list(u:any)	lgth (x), x[i]
fieldlist	
int, posint	arithm op. + relations

Table III Summary of Base CONLAN

References

1. Liskov, Zilles "Specification Techniques for Data Abstractions" IEEE Trans. Software Eng. SE-1, 1, March 75 (pp 7 - 19).
2. Liskov, Zilles "Programming with Abstract Data Types SIGPLAN Notices 9, 4 April 1974 (pp 55-59).
3. Liskov "A design methodology for reliable software systems" Proceedings of AFIPS 41(1972), (191-199).
4. Liskov, Snyder, Atkinson, Shaffert, "Abstraction Mechanisms in CLU," Computations Structures Group Memo 136-1, MIT, January 1977.